Programming assignment
CPE 357
Systems Programming

Assignment 1

Exercise:

Program your own malloc and free functions:
BYTE *mymalloc(int size)
BYTE *myfree(BYTE *addr)
Use brk() and/or sbrk()

MYMALLOC

Use a pagesize of 4096 bytes.
You start with a heapsize of 0, be aware of that.
The mymalloc function needs to do following:
If there is no heap, create one chunk big enough for the requested size with brk/sbrk.
You need to have a structure chunkinfo as part of each memory chunk:

```
struct chunkinfo
        {
        int size; //size of the complete chunk including the chunkinfo
        int inuse; //is it free? 0 or occupied? 1
        BYTE *next,*prev;//address of next and prev chunk
        }
```

Return the correct address for the user-developer!

If there are already chunks present, travers through them as through a list and check:
If the current chunk if free and fits the size, change the chunkinfo accordingly and return the correct address. Split if the chunk is far bigger than the user-developer needs and the rest is bigger than a pagesize.

If no chunk is free or fits, move the program break further: Create a new chunk (and link correctly).

MYFREE

This function sets a chunk to free. Merge adjunct free chunks accordingly.
If the chunk (or the merged chunk) is the last one, you can remove the chunk completely and move the program break back for the size of the chunk.

MUST DO:

- Move the program break back again when the last chunk in the list is removed!
- You analyse function should print the address of the program break!
- Best fit: Search the chunks for the smallest possible chunk which satisfies the request.

Submit:

The whole project folder zipped as filename: FIRSTNAME_LASTNAME_PROGRAM_2.zip on PolyLearn.

How we test (Please put that into your void main):

```
byte*a[100];
analyze();//50% points
for(int i=0;i<100;i++)
       a[i]= mymalloc(1000);
for(int i=0;i<90;i++)
       myfree(a[i]);
analyze(); //50% of points if this is correct
myfree(a[95]);
a[95] = mymalloc(1000);
analyze();//25% points, this new chunk should fill the smaller free one
//(best fit)
for(int i=90;i<100;i++)
       myfree(a[i]);
analyze();// 25% should be an empty heap now with the start address
//from the program start
```

You can use following functions (they come handy)

```
chunkhead* get_last_chunk() //you can change it when you aim for performance
       {
       if(!startofheap) //I have a global void *startofheap = NULL;
              return NULL;
       chunkhead* ch = (chunkhead*)startofheap;
       for (; ch->next; ch = (chunkhead*)ch->next);
       return ch;
       }
void analyze()
       {
       printf("\n-----------------------------------------------------------------\n");
       if(!startofheap)
       {
       printf("no heap");
       return;
       }
       chunkhead* ch = (chunkhead*)startofheap;
       for (int no=0; ch; ch = (chunkhead*)ch->next,no++)
              {
              printf("%d | current addr: %x |", no, ch);
              printf("size: %d | ", ch->size);
              printf("info: %d | ", ch->info);
              printf("next: %x | ", ch->next);
              printf("prev: %x", ch->prev);
              printf("      \n");
              }
       printf("program break on address: %x\n",sbrk(0));
       }
```

Result for comparison

```
--------------------------------------------------------------
no heap, program break on address: 5fff9000
--------------------------------------------------------------
0 | current addr: 5fff9000 |size: 368640 | info: 0 | next: 60053000 | prev: 0
1 | current addr: 60053000 |size: 4096 | info: 1 | next: 60054000 | prev: 5fff9000
2 | current addr: 60054000 |size: 4096 | info: 1 | next: 60055000 | prev: 60053000
3 | current addr: 60055000 |size: 4096 | info: 1 | next: 60056000 | prev: 60054000
4 | current addr: 60056000 |size: 4096 | info: 1 | next: 60057000 | prev: 60055000
5 | current addr: 60057000 |size: 4096 | info: 1 | next: 60058000 | prev: 60056000
6 | current addr: 60058000 |size: 4096 | info: 1 | next: 60059000 | prev: 60057000
7 | current addr: 60059000 |size: 4096 | info: 1 | next: 6005a000 | prev: 60058000
8 | current addr: 6005a000 |size: 4096 | info: 1 | next: 6005b000 | prev: 60059000
9 | current addr: 6005b000 |size: 4096 | info: 1 | next: 6005c000 | prev: 6005a000
10 | current addr: 6005c000 |size: 4096 | info: 1 | next: 0 | prev: 6005b000
program break on address: 6005d000

--------------------------------------------------------------
0 | current addr: 5fff9000 |size: 368640 | info: 0 | next: 60053000 | prev: 0
1 | current addr: 60053000 |size: 4096 | info: 1 | next: 60054000 | prev: 5fff9000
2 | current addr: 60054000 |size: 4096 | info: 1 | next: 60055000 | prev: 60053000
3 | current addr: 60055000 |size: 4096 | info: 1 | next: 60056000 | prev: 60054000
4 | current addr: 60056000 |size: 4096 | info: 1 | next: 60057000 | prev: 60055000
5 | current addr: 60057000 |size: 4096 | info: 1 | next: 60058000 | prev: 60056000
6 | current addr: 60058000 |size: 4096 | info: 1 | next: 60059000 | prev: 60057000
7 | current addr: 60059000 |size: 4096 | info: 1 | next: 6005a000 | prev: 60058000
8 | current addr: 6005a000 |size: 4096 | info: 1 | next: 6005b000 | prev: 60059000
9 | current addr: 6005b000 |size: 4096 | info: 1 | next: 6005c000 | prev: 6005a000
10 | current addr: 6005c000 |size: 4096 | info: 1 | next: 0 | prev: 6005b000
program break on address: 6005d000

--------------------------------------------------------------
no heap, program break on address: 5fff9000
```

Bonus:

We will measure the performance of your code on one and the same computer.
The 5 fastest get 2% bonus on the whole course grade. Be creative!
My results: 521 micro seconds (VirtualBox, Xubuntu, i7-4770, 3.4 GHz)
Test case for speed test (same as above, no print statement except time duration):

```
byte*a[100];
clock_t ca, cb;
for(int i=0;i<100;i++)
       a[i]= mymalloc(1000);
for(int i=0;i<90;i++)
       myfree(a[i]);
              myfree(a[95]);
a[95] = mymalloc(1000);
for(int i=90;i<100;i++)
       myfree(a[i]);
cb = clock();
printf("\nduration: % f\n", (double)(cb - ca);
```

What is allowed:

You can talk and discuss everything on Piazza, helping each other out, but not sharing any code!